Restricting the Flow: Information Bottlenecks for Attribution

Anonymous authors
Paper under double-blind review

Abstract

Attribution methods provide insights into the decision-making of machine learning models like artificial neural networks. For a given input sample, they assign a relevance score to each individual input variable, such as the pixels of an image. In this work we adapt the information bottleneck concept for attribution. By adding noise to intermediate feature maps we restrict the flow of information and can quantify (in bits) how much information image regions provide. We compare our method against ten baselines using three different metrics on VGG-16 and ResNet-50, and find that our methods outperform all baselines in five out of six settings. The method's information-theoretic foundation provides an absolute frame of reference for attribution values (bits) and a guarantee that regions scored close to zero are not required for the network's decision.

1 Introduction

Deep neural networks have become state of the art in many real-world applications. However, their increasing complexity makes it difficult to explain the model's output. For some applications such as in medical decision making or autonomous driving, model interpretability is an important requirement with legal implications. Attribution methods (Selvaraju et al., 2017; Zeiler & Fergus, 2014; Smilkov et al., 2017) aim to explain the model behaviour by assigning a relevance score to each individual input variable. When applied to images, the relevance scores can be visualised as heatmaps over the input pixel space, thus highlighting salient areas relevant for the network's decision.

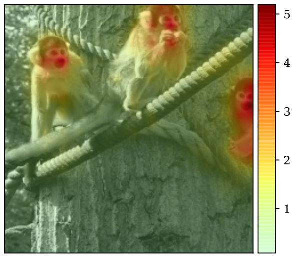


Figure 1: Exemplary heatmap of the Per-Sample Bottleneck.

Attribution maps can be hard to interpret. If an attribution heatmap highlights subjectively irrelevant areas, this might correctly reflect the network's unexpected way of processing the data, or the heatmap might simply be inaccurate. Given an image of a railway locomotive, the attribution map might highlight the train tracks instead of the train itself. Yet, the low relevance values of the locomotive do not guarantee that the network will ignore the locomotive for prediction.

We propose a novel attribution method that estimates the amount of information an image region provides for the network's decision. We use a variational approximation to upper-bound this estimate and therefore can guarantee that areas with zero bits of information are not used. Figure 1 shows an exemplary heatmap of our method. Up to 5 bits per pixel are available for regions corresponding to the monkeys' faces, whereas the tree is scored with close to zero bits per pixel. We can thus guarantee that the tree is not necessary for predicting the correct class, a guarantee to the best of our knowledge other methods can not provide.

To estimate the amount of information, we adapt the information bottleneck concept (Tishby et al., 2000; Alemi et al., 2017). The bottleneck is inserted into an existing neural network and restricts the information flow by adding noise to the activation maps. Unimportant activations are replaced almost entirely by noise, removing all information for subsequent network layers. We developed two approaches to learn the parameters of the bottleneck – either using a single sample (Per-Sample Bottleneck), or the entire dataset (Readout Bottleneck).

We evaluate against ten different baselines using three metrics. First, we calculated the Sensitivity-n metric proposed by Ancona et al. (2018). Secondly, we quantified how well the object of interest was localized using bounding boxes and extend the degradation task proposed by Ancona et al. (2017). In all these metrics our method outperforms the baselines consistently. For reproducibility, we share our source code*. Our method provides a theoretic upper-bound on the used information while demonstrating strong empirical performance. This contributes to improving model explainability and to increasing trust in the attribution results.

To summarize our contributions:

- We adapt the information bottleneck concept for attribution to estimate the information used by the network. Information theory provides a guarantee that areas scored irrelevant are indeed not neccessary for the network's prediction.
- We propose two ways – *Per-Sample* and *Readout* Bottleneck – to learn the parameters of the information bottleneck.
- We contribute a novel evaluation method for attribution based on bounding boxes and we also extend the metric proposed by Ancona et al. (2017) to provide a single scalar value and improve the metric's comparability between different network architectures.

2 Related Work

Attribution is an active research topic. *Gradient Maps* (Baehrens et al., 2010) and *Saliency Maps* (Simonyan & Zisserman, 2014) are based on calculating the gradient of the target output neuron w.r.t. to the input features. *Integrated Gradient* (Sundararajan et al., 2017) and *SmoothGrad* (Smilkov et al., 2017) improve over gradient-based attribution maps by averaging the gradient of multiple inputs, either in a local neighborhood or over brightness level interpolations of the input image. Other methods, such as *Layer-wise Relevance Propagation (LRP)* (Bach et al., 2015), *Deep Taylor Decomposition (DTD)* (Montavon et al., 2017), *Guided Backpropagation (GuidedBP)* (Springenberg et al., 2014) or *DeepLIFT* (Shrikumar et al., 2017) modify the propagation rule. *PatternAttribution* (Kindermans et al., 2018) builds upon LRP by estimating the signal's direction for the backward propagation. Perturbation-based methods are not based on backpropagation and treat the model as a black-box. *Occlusion* (Zeiler & Fergus, 2014) measures the importance of individual features by replacing them and then measuring the drop in classification score. *Grad-Cam* (Selvaraju et al., 2017) take the activations of the final convolutional layer to compute relevance scores. They also propose combine thier method with GuidedBP called *GuidedGrad-CAM*. Ribeiro et al. (2016) use image superpixels to explain deep neural networks. To our the best of our knowledge, we are the first to estimate the amount of used information for attribution purposes.

Although many attribution methods exist, no common evaluation benchmark is established. Thus, determining the state of the art is difficult. The performance of attribution methods is highly dependent on the used model and dataset. Often only a pure visual comparison is performed (Smilkov et al., 2017; Springenberg et al., 2014; Montavon et al., 2017; Sundararajan et al., 2017; Bach et al., 2015). The most commonly used benchmark is the degragation score (Kindermans et al., 2018; Samek et al., 2016; Ancona et al., 2017). Ancona et al. (2018) propose the Sensitivity-n score based on the correlation between degragation and model performance. For the ROAR score (Hooker et al., 2018), the network is trained from scratch on the degraded images. While computationally expensive, it ensures the change in accuracy does not stem from out-of-domain inputs – an inherent problem of masking.

Adding noise to a signal reduces the amount of information (Shannon, 1948). It is therefore a popular way to regularize neural networks (Srivastava et al., 2014; Kingma et al., 2015; Gal et al., 2017). However, for regularization, the noise is applied independently from the input and therefore no attribution maps can be obtained. In Variational Autoencoders (VAEs) (Kingma & Welling, 2013), noise is used to build an information bottleneck that restricts the information capacity of the latent code. In our work, we construct a similar information bottleneck that can be inserted into an existing network. Deep convolutional neural networks have been augmented with information bottlenecks before to improve the generalization and robustness against adversarial examples (Achille & Soatto, 2018; Alemi et al., 2017).

[*] https://github.com/attribution-bottleneck/attribution-bottleneck-pytorch

3 Information Bottleneck for Attribution

Instead of a backpropagation approach, we will quantify the flow of information through the network in the *forward* pass. Given a pre-trained model, we inject noise into a feature map, which restrains the flow of information through it. We optimize the intensity of the noise to minimize the information flow, while simultaneously maximizing the original model objective, the classification score. The parameters of the original model are not changed.

3.1 Information Bottleneck

Generally, the information bottleneck concept (Tishby et al., 2000) describes a limitation of available information. Usually, the labels Y are predicted using all information from the input X. The information bottleneck limits the information to predict Y by introducing a new variable Z. The information bottleneck then maximizes the information Z shares about the labels Y while minimizing the information Z contains about X:

$$\max I[Y;Z] - \beta I[X,Z] \;, \tag{1}$$

where $I[X,Z]$ denotes the mutual information and β controls the trade-off between predicting the labels well and using little information of X. A common way to reduce the amount of information is to add noise (Alemi et al., 2017; Kingma & Welling, 2013).

For attribution, we inject an information bottleneck into a pretrained network. The bottleneck is inserted into an early layer to ensure that the information in the network is still local, e.g. for the ResNet the bottleneck is added after the second ResNet block. In our case, X denote the feature map at this specific depth of the network. We want to reduce information in X by adding noise. Yet, as the neural network is already trained, we have to preserve the magnitude and variance of the input to the following layers. Therefore, we also damp the signal X when increasing the noise, effectively replacing the signal partly with noise. In the extreme case, when no signal is transmitted, we replace X completely with noise of the same mean and variance as X. For this purpose, we estimate the mean μ_X and variance σ_X^2 of each feature of X empirically over the training data. As information bottleneck, we then apply a linear interpolation between signal and noise:

$$Z = \lambda(X)X + (1 - \lambda(X))\,\epsilon \;, \tag{2}$$

where $\epsilon \sim \mathcal{N}(\mu_X, \sigma_X^2)$ and $\lambda(X)$ controls the damping of the signal and addition of the noise. The value of λ is a tensor with the same dimensions as X. Given $\lambda_i(X) = 1$ at the feature map location i, the bottleneck transmits all information as $Z_i = X_i$. Whereas if $\lambda_i = 0$, then $Z_i = \epsilon$ and thus all information of X_i is lost and replaced with noise. It could be tempting to think that Z from equation 2 has the same mean and variance as X. This is not the case in general as $\lambda(X)$ depends on X (an analysis of this is shown in the appendix E).

In our method, we consider an area relevant if it contains useful information for classification. We therefore need to estimate how much information Z still contains about X. This quantity is the mutual information $\mathrm{I}[X,Z]$ that can be written as:

$$\mathrm{I}[X,Z] = \mathbb{E}_X[D_{\mathrm{KL}}[P(Z|X)||P(Z)]] \,, \tag{3}$$

where $P(Z|X)$ and $P(Z)$ denote the respective probability distributions. We have no analytic expression for $P(Z)$ since it would be necessary to integrate over the data $p(z) = \int_X p(z|x)p(x)\mathrm{d}x$ – an intractable integral. It is a common problem that the mutual information can not be computed exactly but is rather approximated (Poole et al., 2019; Suzuki et al., 2008). We resort to a variational approximation $Q(Z) = \mathcal{N}(\mu_X, \sigma_X)$ which assumes that all dimensions of Z are normally distributed and independent, a reasonable assumption, as activations after linear or convolutional layers tend to have a Gaussian distribution (Klambauer et al., 2017). The independence assumption will generally not hold. However, this will only lead to an overestimation of the mutual information as shown below. Substituting $Q(Z)$ into the previous equation 3, we obtain:

$$\mathrm{I}[X,Z] = \mathbb{E}_X[D_{\mathrm{KL}}[P(Z|X)||Q(Z)]] - D_{\mathrm{KL}}[Q(Z)||P(Z)] \,. \tag{4}$$

The derivation is shown in appendix D and follows the one of Alemi et al. (2017). The first term contains the KL-divergence between two normal distributions and can therefore be evaluated easily.

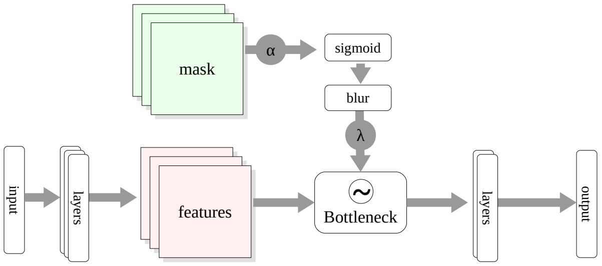


Figure 2: Principle of the *Per-Sample Bottleneck*. The mask (green) contains an α_i for each x_i in the intermediate feature maps (red). Depending on α, more or less information is passed on to the next layer. The mask is iteratively optimized according to equation 6 for each sample individually.

We will use the first KL-divergence to approximate the mutual information. The information loss function $\mathcal{L}_I$ is therefore:

$$\mathcal{L}_I = \mathbb{E}_X[D_{\mathrm{KL}}[P(Z|X)||Q(Z)]] \,. \tag{5}$$

We know that $\mathcal{L}_I$ overestimates the mutual information, i.e. $\mathcal{L}_I \geq \mathrm{I}[X, Z]$ as the second KL-divergence term $D_{\mathrm{KL}}[Q(Z)||P(Z)]$ has to be positive. If $\mathcal{L}_I$ is zero for an area, we can guarantee that no information from this area is used for prediction.

We aim to only keep the information needed for correct classification. Thus, the mutual information should be minimal while the classification score should remain high. Let $\mathcal{L}_{CE}$ be the cross-entropy of the classification. Then, we obtain the following optimization problem:

$$\mathcal{L} = \mathcal{L}_{CE} + \beta\mathcal{L}_I \,, \tag{6}$$

where the parameter β controls the relative importance of both objectives. A small β will result in more bits of information flowing, whereas a higher β will result in more information getting discarded.

We propose two ways of finding the parameters λ – the Per-Sample Bottleneck and the Readout Bottleneck. For the Per-Sample Bottleneck, we optimize λ for each image individually, whereas in the readout bottleneck, we train a distinct neural network to predict λ.

3.2 Per-Sample Bottleneck

For the *Per-Sample Bottleneck*, we use the bottleneck formulation described above and optimize $\mathcal{L}$ for individual samples – not for the complete dataset at once. Given a sample x, λ is fitted to the sample to reflect important and unimportant regions in the feature space. A diagram of the Per-Sample Bottleneck is shown in Figure 2.

Parameterization: The bottleneck parameters λ have to be in $[0, 1]$. To simplify optimization, we parametrize $\lambda = \mathrm{sigmoid}(\alpha)$. This allows α to be chosen freely as $\alpha \in \mathbb{R}^d$ without having to explicitly clamp λ to $[0, 1]$ during optimization.

Initialization: For training neural networks, the initialization of parameters matters and has a strong impact on the training output. In the beginning, we want that all information is retained. Therefore for all dimensions i, we initialize $\alpha_i = 5$ and thus $\lambda_i \approx 0.993 \Rightarrow Z \approx X$. At first, The bottleneck has no practical impact on the model performance. It then deviates from this starting point to suppress unimportant regions.

Optimization: We use a fixed number of 10 iterations using the Adam optimizer (Kingma & Ba, 2014) with learning rate 1 to train the mask α. Although we optimize on a single sample, we use a minibatch of size 10 with different noise tensors added to the sample to stabilize the training. In total, we execute the model 100 times to create a heatmap, comparable to other methods such as SmoothGrad. After the optimization, the model usually predicts the target with probability close to 1 indicating that all negative evidence was removed.

Measure of information: For a measure of importance of each feature in Z, we can simply evaluate $D_{\mathrm{KL}}(P(Z|X)||Q(Z))$ per dimension: It measures where in the network the information flows and is

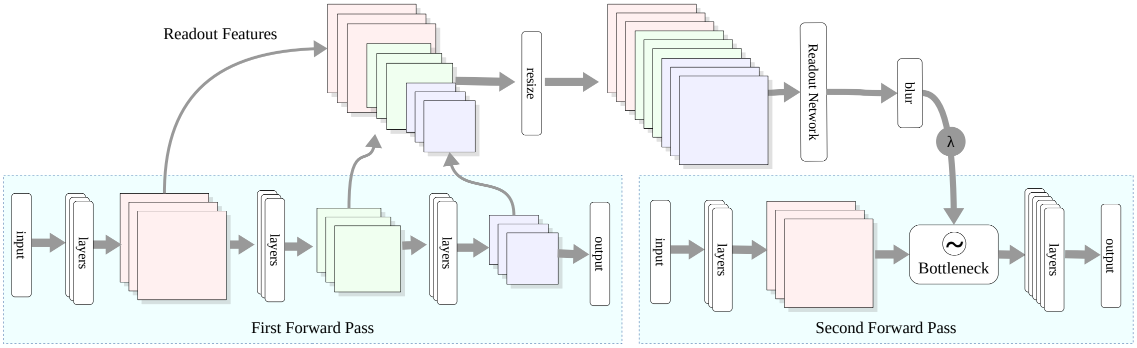


Figure 3: Principle of the *Readout Bottleneck*. In the first forward pass, feature maps are collected at different depths. The readout network uses a scaled version of the feature maps to predict the parameters for the bottleneck layer. In the second forward pass, the bottleneck is inserted and noise added. All parameters of the analyzed network are kept fixed.

less suppressed after training. To obtain a two-dimensional heatmap m, we sum over the channel axis: $m_{[h,w]} = \sum_{i=0}^{c} D_{\mathrm{KL}}(P(Z_{[i,h,w]}|X_{[i,h,w]})||Q(Z_{[i,h,w]}))$. As convolutional neural networks preserve the locality in their channel maps, we use bilinear interpolation to match the map to the spatial dimensions of the input. The bottleneck is inserted into an early layer to ensure that the information in the network is still local, e.g. for the ResNet the bottleneck is added after the second ResNet block. Choosing a later layer with lower spatial resolution would also increase the blurriness of the attribution maps due to the required interpolation.

Enforcing local smoothness: Pooling operations and convolutional layers with stride greater than 1 are ignoring parts of the input. This causes the Per-Sample Bottleneck to overfit to a grid structure as shown in Appendix C. To obtain a robust and smooth attribution map, we convolve the sigmoid output with a fixed Gaussian kernel with standard deviation σ_s. Smoothing the mask during training is *not* equivalent to smoothing the resulting attribution map, as during training also the gradient is averaged locally. The parametrization for the Per-Sample Bottleneck is:

$$\lambda = \mathrm{blur}(\sigma_s, \mathrm{sigmoid}(\alpha)) \ . \tag{7}$$

3.3 Readout Bottleneck

In this section, we train a second neural network to predict the mask α. In contrast to the Per-Sample Bottleneck, this model is trained on the entire training set. In Figure 3, the Readout Bottleneck is depicted.

The readout concept was first introduced for gaze prediction by Kümmerer et al. (2014). Similarly, we collect feature map values from different depths in a first forward pass without adding any noise. As the spatial resolution of the feature maps differ, we interpolate them bilinearly to match the spatial dimensions of the bottleneck layer. The readout network then predicts the information mask based on the collected feature maps. In a second forward pass, we insert the bottleneck layer into the network and restrict the information flow.

Except from this formulation as a function of the readout values, the Readout Bottleneck is identical to the mechanism of the Per-Sample Bottleneck. The measure of information works in the same way as for the Per-Sample Bottleneck and we also use the same smoothing. Given a new sample, we can obtain a heatmap simply by collecting the feature maps and executing the readout network.

The readout network consists of three 1x1 convolutional layers. ReLU activations are applied between convolutional layers and a final sigmoid activation yields $\lambda \in [0, 1]$. As the input includes the upscaled feature maps, the field-of-view is large although the network itself only has 1x1 kernels.

Table 1: Influence of β on the information loss $\mathcal{L}_I$ and the test accuracy. k is the size of the feature map, i.e. $k = hwc$. A Readout Bottleneck is inserted into a ResNet-50. *Initial*: Configuration of the untrained bottleneck with $\alpha = 5$. *Original*: Values for the original model without the bottleneck.

	Original	Initial	$\beta = 0.01/k$	$0.1/k$	$1/k$	$10/k$	$100/k$
$\mathcal{L}_I/k$	–	2.500	1.822	0.628	0.222	0.079	0.023
Top-5 Accuracy	0.928	0.928	0.930	0.928	0.917	0.870	0.505
Top-1 Accuracy	0.760	0.760	0.761	0.756	0.735	0.660	0.302

4 Evaluation

4.1 Experimental Setup

As neural network architectures, we selected the ResNet-50 (He et al., 2016) and the VGG-16 Simonyan & Zisserman (2014), using pretrained weights from the torchvision package (Paszke et al., 2017; Marcel & Rodriguez, 2010). These two models cover a range of concepts: Different dimensionality reduction methods (by stride, max-pooling, average-pooling), residual connections, batch normalization, dropout, low depth (16-weight-layer VGG) and high depth (50-weight-layer ResNet). This variety makes the evaluation less likely to overfit on a specific model type. They are commonly used in literature concerning attribution methods. For PatternAttribution on the VGG-16, we obtained weights for the signal estimators from Kindermans et al. (2018).

As naive baselines, we selected random attribution, Occlusion with patch sizes 8x8 and 14x14, and Gradient Maps. SmoothGrad and Integrated Gradients cover methods that accumulate gradients. With PatternAttribution, GuidedBP, and LRP, we include three methods with a modified backpropagation rule. As PatternAttribution and LRP do not support skip connections, we report no results for them on the ResNet-50. We also include Grad-CAM and its combination with GuidedBP, GuidedGrad-CAM.

For the compared methods, we use the hyperparameters suggested by the original authors with one exception. For LRP (Bach et al., 2015), the original hyperparameters produced unsatisfactory results. With $\epsilon = 5$ and $\beta = -1$, we found LRP to yield best visual results. For our methods, the hyperparameters are obtained using grid search with the degradation metric as objective (Appendix B). The readout network is trained over the training set of the ILSVRC12 dataset (Russakovsky et al., 2015) for $E = 30$ epochs, i.e. the same dataset the original models were trained on.

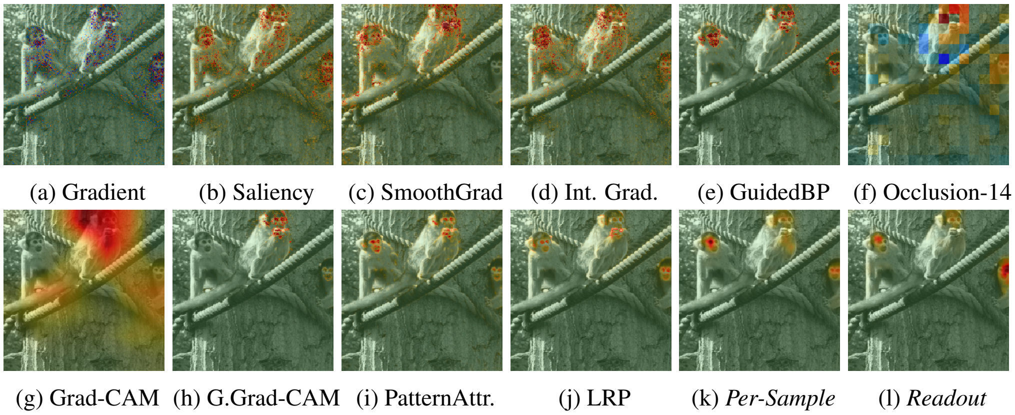
The optimization objective of the bottleneck is $\mathcal{L}_{CE} + \beta\mathcal{L}_I$ as given in equation 6. Generally, the information loss $\mathcal{L}_I$ is larger than the classifier loss by several orders of magnitude as it sums over height h, width w and channels c. We therefore use $k = hwc$ as a reference point to select β in the range from $0.01/k$ to $100/k$. A comparison of pre- to post-training accuracy and the estimated mutual information is shown in Table 1. Notably, for a small $\beta \leq 0.1/k$, the bottleneck even improves the final accuracy slightly as negative evidence in the image can be removed. Naturally, as β is raised, less information is transmitted per feature, resulting in a lower accuracy. We evaluate the Per-Sample bottleneck for $\beta = 1/k, 10/k, 100/k$. The Readout network is trained with the best performing value $\beta = 10/k$.

4.2 Qualitative Assessment

In Figure 4, the heatmaps of all evaluated samples are shown. More samples are shown in Appendix A. Subjectively, both the Per-Sample and Readout Bottleneck identify well areas relevant for the classification. While Guided Backpropagation and PatternAttribution tend to highlight edges, the Per-Sample Bottleneck focuses on image regions. For the Readout Bottleneck, the attribution is concentrated a little more on the object edges. Compared to Grad-CAM, both our methods are more specific, i.e. less pixels are scored high.

4.3 Bounding Box

To quantify how well attribution methods identify and localize the object of interest, we rely on bounding boxes available for the ImageNet dataset. Bounding boxes may contain irrelevant areas especially for non-rectangular objects. We restrict our evaluation to images with bounding boxes


(a) Gradient (b) Saliency (c) SmoothGrad (d) Int. Grad. (e) GuidedBP (f) Occlusion-14

(g) Grad-CAM (h) G.Grad-CAM (i) PatternAttr. (j) LRP (k) *Per-Sample* (l) *Readout*

Figure 4: Heatmaps of all implemented methods for the VGG-16 (see Appendix A for more).

covering less than 33% of the input image. In total, we run the bounding box evaluation on 11,849 images from the ImageNet validation set.

If the bounding box contains n pixels, we measure how many of the n-th highest scored pixels are contained in the bounding box. By dividing by n, we obtain a ratio between 0 and 1. In Table 2, the results are shown on the right under the *bbox* task. The Per-Sample Bottleneck outperforms all other methods on VGG-16 with a margin of 12.9% and on ResNet-50 with 15.1%.

The attribution maps produced by the other methods each differ in their distribution of attribution scores. Gradient-based methods such as GuidedBP and LRP tend to produce rather sparse attribution maps, while Grad-CAM and our method yield smoother, dense maps. Sparse methods assign the bulk of attribution mass to only a few image regions. Since we regard only the top n scored pixels, we may sample regions outside the bounding box from the remaining low-attribution tail, since even infinitesimal score differences matter in the ordering process. An alternative metric would be to take the sum of attribution in the bounding box and compare it to the total attribution in the image. We found this metric is not robust against extreme values. For the ResNet-50, we found basic Gradient Maps to be the best method as a few pixels receiving extrem scores are enough to dominate the sum.

4.4 Sensitivity-n

Ancona et al. (2018) proposed Sensitivity-n as evaluation metric for attribution methods. Sensitivity-n masks the network's input randomly and then measures how strong the amount of attribution in the mask correlates with the drop in classifier score. Given a set T_n containing n randomly selected pixel indices, Sensitivity-n measures the Pearson correlation coefficient:

$$\operatorname{corr}\left(\sum_{i \in T_n} R_i(x),\ S_c(x) - S_c(x_{[x_{T_n}=0]})\right), \tag{8}$$

where $S_c(x)$ is the classifier logit output for class c, R_i is the relevance at pixel i and $x_{[x_{T_n}=0]}$ denotes the input with all pixels in T_n set to zero. As in the original paper, we pick the number of masked pixels n in logscale between 1 and 80% of all pixels. For each n, we generate 100 different index sets T and test each on 1000 randomly selected images from the validation set. The correlation is calculated over the different index sets and then averaged over all images.

In Figure 5, the Sensitivity-n scores are shown for ResNet-50 and VGG-16. When masking inputs pixel-wise as in Ancona et al. (2018), the Sensitivity-n score across methods are not well discriminable, all scores range within the lower 10% of the scale. We therefore ran the metric with a tile size of 8x8 pixels. Occlusion-8x8 performs perfectly as its relevance scores correspond directly to the drop in logits per 8x8 tile. For all other baselines, we find that the Readout Bottleneck dominates the field until $n = 10^3$ pixels and that the Per-Sample Bottlenecks with $\beta = 10/k, 100/k$ perform best above $n = 10^3$ pixels.

Table 2: *Degradation (deg.)*: Integral between LeRF and MoRF in the degradation benchmark for different models and window sizes over the ImageNet test set. *Bounding Box (bbox)*: ratio of the highest scored pixels within the bounding box. PatternAttribution and LRP do not support ResNet-50.

Model	ResNet-50	ResNet-50	VGG-16	ResNet-50	VGG-16
Task	deg. 8x8	deg. 14x14	deg. 14x14	bbox	bbox
Random	0.000	0.000	0.000	0.167	0.167
Occlusion-8x8	0.162	0.130	0.256	0.296	0.312
Occlusion-14x14	0.228	0.231	0.399	0.341	0.358
Gradient	0.002	0.005	0.004	0.259	0.276
Saliency	0.287	0.305	0.357	0.363	0.393
GuidedBP	0.492	0.516	0.490	0.388	0.373
PatternAttribution	–	–	0.454	–	0.404
LRP	–	–	0.464	–	0.441
Int. Grad. (of Saliency)	0.401	0.424	0.448	0.373	0.396
SmoothGrad (of Saliency)	0.486	0.502	0.451	0.439	0.399
Grad-CAM	0.536	0.541	0.514	0.465	0.399
GuidedGrad-CAM	0.565	0.577	**0.573**	0.467	0.418
Per-Sample $\beta = 1/k$	0.558	0.563	0.557	0.584	0.369
Per-Sample $\beta = 10/k$	**0.585**	**0.586**	0.570	**0.618**	0.536
Per-Sample $\beta = 100/k$	0.584	0.584	0.548	**0.618**	**0.570**
Readout $\beta = 10/k$	0.526	0.526	0.500	0.483	0.482

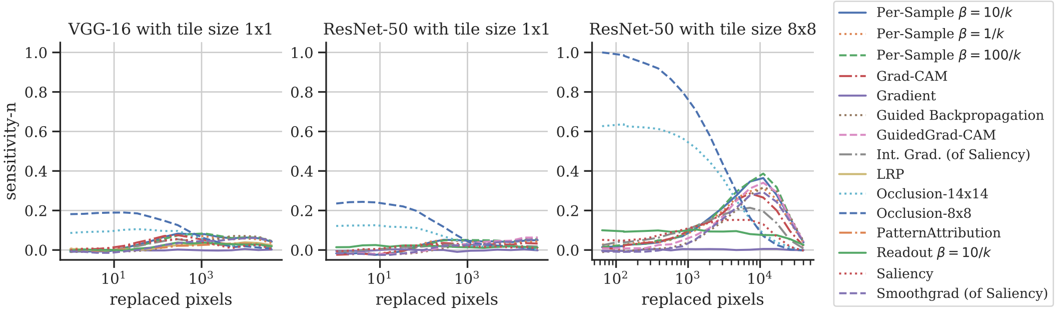


Figure 5: Sensitivity-n scores for ResNet-50 and VGG-16. For the ResNet-50, also tile size 8x8 is shown. Best viewed in color.

4.5 Image Degradation

As a further quantitative evaluation, we rely on the degradation task as used by Ancona et al. (2017); Kindermans et al. (2018); Hooker et al. (2018); Samek et al. (2016). Given an attribution heatmap, the input is split in tiles which are ranked by the sum of attribution values within each corresponding tile of the attribution. At each iteration, the highest ranked tile is replaced with a constant value, the modified input is fed through the network, and the resulting drop in target class score is measured. A steep descent of the accuracy curve indicates a meaningful attribution map.

When implemented in the described way, the most relevant tiles are removed first (MoRF). However, Ancona et al. (2017) argue that using only the MoRF curve for evaluation is not sufficient. For the MoRF score, it is beneficial to find tiles that disrupt the output of the neural network as early as possible in the sequence of tiles. Neural networks have been shown to be sensitive to subtle changes in the input (Szegedy et al., 2013). Hence, the tiles do not necessarily have to contain meaningful information to disrupt the network. Therefore, Ancona et al. (2017) proposes to invert the degradation direction, removing tiles ranked as least relevant by the attribution method first (LeRF). The LeRF task favors methods that identify areas sufficient for classification.

Ancona et al. (2017) measures the drop in target class logits, which may vary in scale for different models and does not account for scores of other classes. To improve the comparability between different models, we measure the drop in final target class probability. Furthermore, we scale the model output probability $p(y|x)$ such that it becomes independent from the model performance. The scaled probability $s(x)$ is then:

$$s(x) = \frac{p(y|x) - b}{t_1 - b} \ , \tag{9}$$

where t_1 is the model's average top-1 probability on the original samples and b is the mean model output on the fully degradad images. Both averages are taken over the validation set. A score of 1 corresponds to the original model performance. The score $s(x)$ still depends on the model, as every model behaves differently on degraded images. Our extension of Ancona et al. (2017) is a re-scaling and thus does not change the ordering of the attribution methods.

Both LeRF and MoRF degradation yield curves as visualized in Figure 6, measuring different qualities of the attribution method. In order to obtain a scalar measure of attribution performance and to combine both metrics, we propose to calculate the integral between the MoRF and LeRF curves. This scalar can then be interpreted as the average difference of the drop in model accuracy when replacing most important and least important tiles first, respectively.

The results for all implemented attribution methods on the degradation task are given in Table 2. We evaluated both models on 14x14 tiles. For the ResNet-50, we additonally included 8x8 tiles. For the VGG-16, we randomly choose 10000 samples from the validation set. The ResNet is evaluated on the full validation set. We show the mean LeRF and MoRF curves of all methods in Appendix F. The Per-Sample Bottleneck outperforms all other methods in the degradation benchmark except for GuidedGrad-CAM on VGG-16 where it scores comparably (score difference of 0.003). The Readout Bottleneck achieves a generally lower degradation scores but still perform competitively.

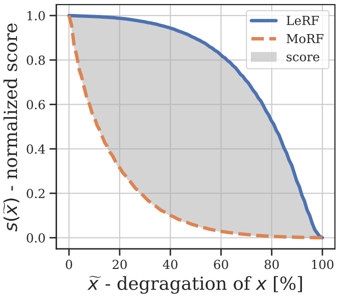


Figure 6: Mean MoRF and LeRF for the Per-Sample Bottleneck. The area between both curves is the final degradation score.

5 Conclusion

We propose two novel attribution methods that return an upper bound on the amount of information each input region provides for the network's decision. Our models' core functionality is a bottleneck layer used to inject noise into a given feature layer and a mechanism to learn the parametrized amount of noise per feature. The Per-Sample Bottleneck is optimized per single data point, whereas the Readout Bottleneck is trained on the entire dataset.

Our method does not constrain the internal network structure. In contrast to several backpropagation-based methods, it supports any activation function and network architecture. To evaluate our method we proposed a novel variant of the degradation task to quantify model performance deterioration when removing both relevant and irrelevant image tiles first. Our Per-Sample Bottleneck and Readout Bottle both show competitive results on all metrics used, outperforming the state of the art with significant margin for some of the tasks.

The method's information-theoretic foundation provides a guarantee that the network does not use regions of zero-valued attribution. To our knowledge, our attribution methods is the only one to provide scores with units (bits). This absolute frame of reference allows a quantitative comparisons between models, inputs and input regions. We hope this contributes to a deeper understanding of neural networks and creates trust to use modern models in sensitive areas of application.

Acknowledgement

Hidden for double-blind review.

REFERENCES

Alessandro Achille and Stefano Soatto. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40 (12):2897–2905, 2018.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations (ICLR 2017)*, 2017.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. In *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*. ETH Zurich, 2017.

Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7), 2015.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating Feature Importance Estimates. *arXiv e-prints*, 2018.

Pieter-Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. In *International Conference on Learning Representations*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pp. 971–980, 2017.

Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. *arXiv:1411.1045 [cs, q-bio, stat]*, 2014.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-933-6.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 2019.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153. JMLR.org, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv:1706.03825 [cs, stat]*, 2017.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv e-prints*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating mutual information by maximum likelihood density ratio estimation. In *New challenges for feature selection in data mining and knowledge discovery*, pp. 5–20, 2008.

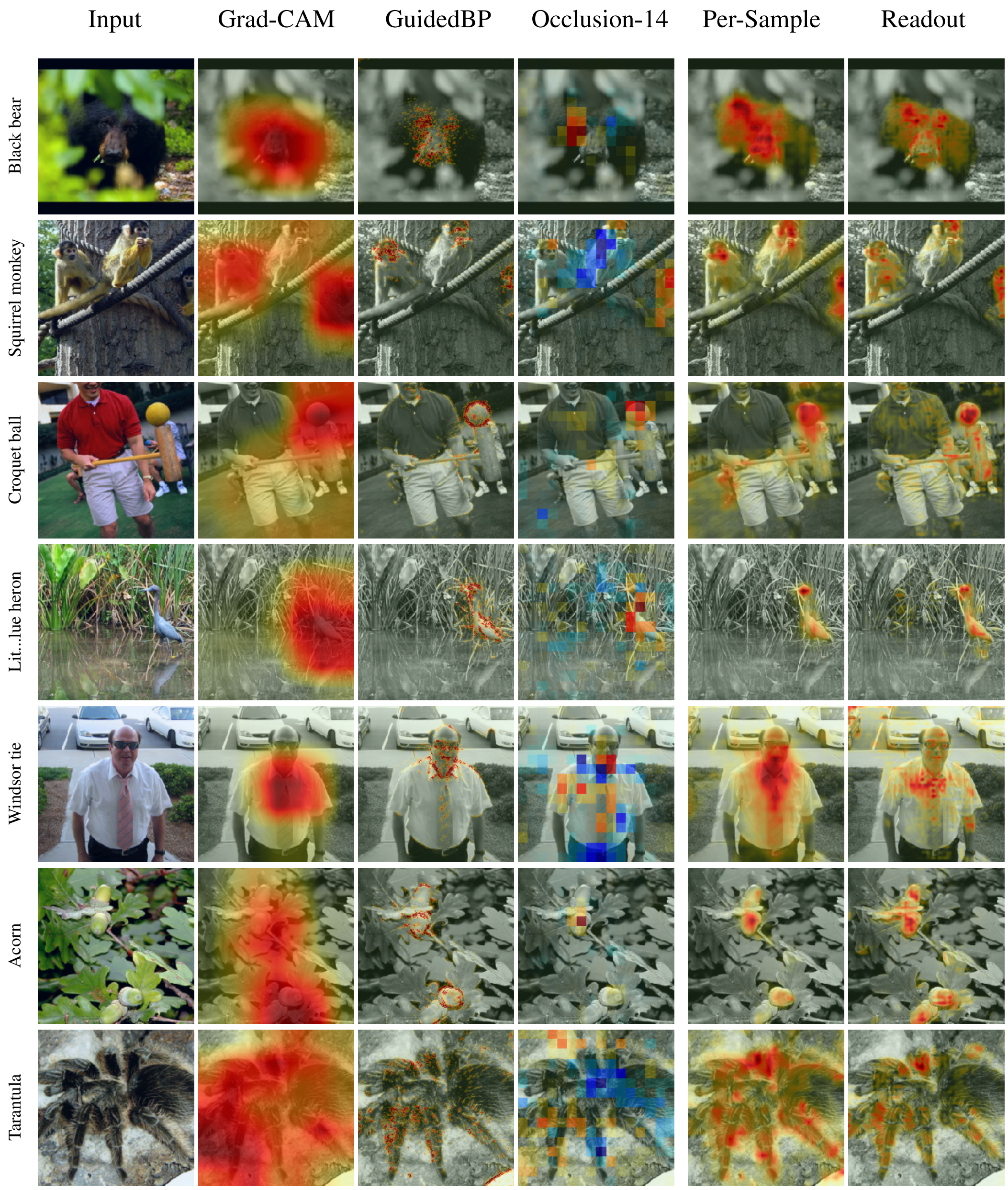
Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

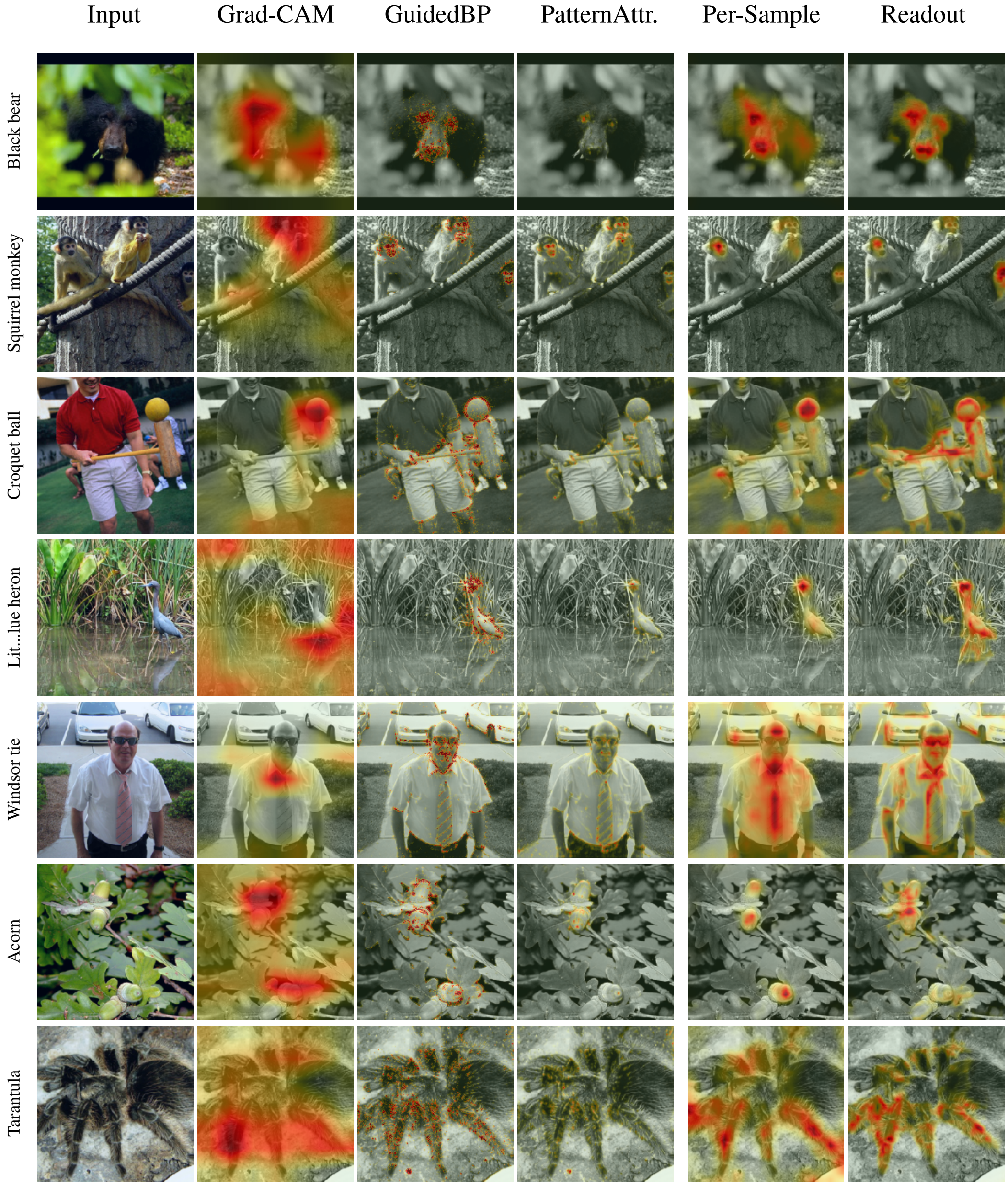
Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

A Visual Comparison of Attribution Methods

A.1 ResNet-50



A.2 VGG-16


Input
Grad-CAM
GuidedBP
PatternAttr.
Per-Sample
Readout
Black bear
Squirrel monkey
Croquet ball
Lit...lue heron
Windsor tie
Acorn
Tarantula

B Hyperparameters

Parameter	ResNet-50	VGG-16	Search space
Target layer	`conv2_4`	`conv_8`	
Optimizer	Adam (Kingma & Ba (2014))		
Learning Rate	$\eta = 1$		$\{0.03, 0.1, 0.3, 1, 3, 10\}$
Balance Factor	$\beta = 10/k$		$\{0.001, 0.01, 0.1, 1, 10, 100, 300\}$
Iterations	$T = 10$		$\{1, 3, 5, 10, 30, 100\}$
Batch Size	$B = 10$		$\{1, 5, 10, 30\}$
Smoothing	$\sigma_s = 1$		$\{0.5, 1, 2\}$

Table 3: Hyperparameters for Per-Sample Bottleneck. The layer notations for the ResNet-50 are taken from the original publication (He et al., 2016). The first index denotes the block and the second the layer within the block. For the VGG-16, `conv_n` denotes the n-th convolutional layer.

Parameter	ResNet-50	VGG-16	Search space
Target layer	`conv1_3`	`conv_5`	
Reading out	`conv1_3`	`conv_5`	
	`conv2_4`	`conv_8`	
	`conv3_6`	`conv_11`	
	`conv4_3`	`conv_13`	
	`fc`	`fc`	
Optimizer	Adam (Kingma & Ba (2014))		
Learning Rate	$\eta = 10^{-5}$		{e-4, e-5, e-6}
Balance Factor	$\beta = 10/k$		$\{0.1/k, 1/k, 10/k, 100/k\}$
Epochs	$E = 10$		
Batch Size	$B = 16$		
Smoothing	$\sigma_s = 1$		

Table 4: Hyperparameters for the Readout Bottleneck.

C Grid Artifacts when not using Smoothing

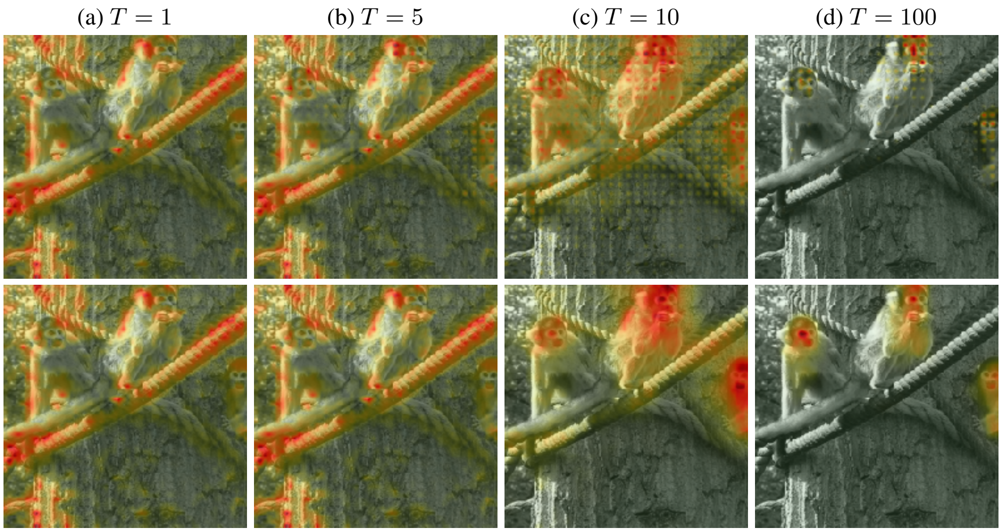


Figure 7: Development of $D_{\mathrm{KL}}(Q(Z|X)||Q(Z))$ for layer `conv1_3` of the ResNet-50. Red indicate areas with maximal information flow and semi-transparent green for zero information flow. Top row: without smoothing the mask exhibits a grid structure. Bottom row: smoothing with $\sigma_s = 2$. The smoothing both prevents the artifacts and reduces overfitting to small areas.

D Derivation of the Approximation of Mutual Information

For the mutual information $\mathrm{I}[X, Z]$, we have:

$$\begin{aligned}
\mathrm{I}[X, Z] &= \mathbb{E}_X[D_{\mathrm{KL}}[P(Z|X)||P(Z)]] && (10)\\
&= \int_X p(x) \left(\int_Z p(x)p(z|x) \log \frac{p(z|x)}{p(z)} dz \right) dx && (11)\\
&= \int_X \int_Z p(x,z) \log \frac{p(z|x)}{p(z)} \frac{q(z)}{q(z)} dz dx && (12)\\
&= \int_X \int_Z p(x,z) \log \frac{p(z|x)}{q(z)} dz dx + \int_X \int_Z p(x,z) \log \frac{q(z)}{p(z)} dz dx && (13)\\
&= \int_X \int_Z p(x,z) \log \frac{p(z|x)}{q(z)} dz dx + \int_Z p(z) \left(\int_X p(x|z) dx \right) \log \frac{q(z)}{p(z)} dz && (14)\\
&= \mathbb{E}_X \left[D_{\mathrm{KL}}[P(Z|X)||Q(Z)] \right] - D_{\mathrm{KL}}[Q(Z)||P(Z)] && (15)
\end{aligned}$$

E Mean and variance of Z

The $\lambda(X)$ linearly interpolate between the feature map X and the noise $\epsilon \sim \mathcal{N}(\mu_X, \sigma_X^2)$, where μ_X and σ are the estimated mean and standard derivation of X.

$$Z = \lambda(X)X + (1 - \lambda(x))\epsilon \tag{16}$$

For the mean of Z, we have:

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}[\lambda(X)X] + \mathbb{E}[(1 - \lambda(X))\epsilon] && \triangleright \text{ substituting in definition of } Z\\
&= E[\lambda(X)X] + \mathbb{E}[1 - \lambda(X)]\mathbb{E}[\epsilon] && \triangleright \text{ independence of } \lambda \text{ and } \epsilon\\
&= \mathrm{cov}(\lambda(X), X) + \mathbb{E}[\lambda(X)]\mathbb{E}[X] + \mathbb{E}[1 - \lambda(X)]\mathbb{E}[\epsilon] && \triangleright \mathrm{cov}(A, B) = \mathbb{E}[AB] - \mathbb{E}[A]\mathbb{E}[B]\\
&\approx \mathrm{cov}(\lambda(X), X) + \mathbb{E}[\epsilon] && \triangleright \mathbb{E}[\epsilon] = \mu_X \approx \mathbb{E}[X]
\end{aligned}$$

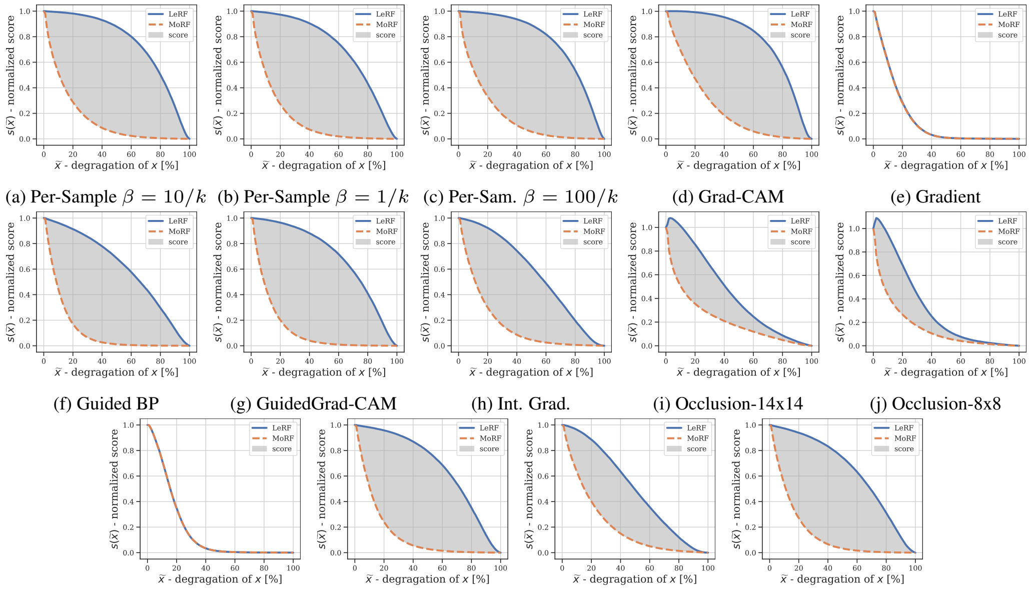
As $\lambda(X)$ and X are multipled together, they form a complex product distribution. If they do not correlate, $\mathbb{E}[Z] \approx E[\epsilon] \approx E[X]$.

A similar problem araises for the variance:

$$\begin{aligned}\mathrm{Var}[Z] &= \mathbb{E}[Z^2] - \mathbb{E}[Z]^2 \\ &= \mathbb{E}[(\lambda(X)X + (1-\lambda(X)\epsilon)^2] - (\mathrm{cov}(\lambda(X), X) + \mathbb{E}[\epsilon])^2\end{aligned}$$

The multiplication of $\lambda(X)$ and X causes in general the variance of Z and X to not match: $\mathrm{Var}[Z] \neq \mathrm{Var}[X]$

F MoRF and LeRF Degradation Paths

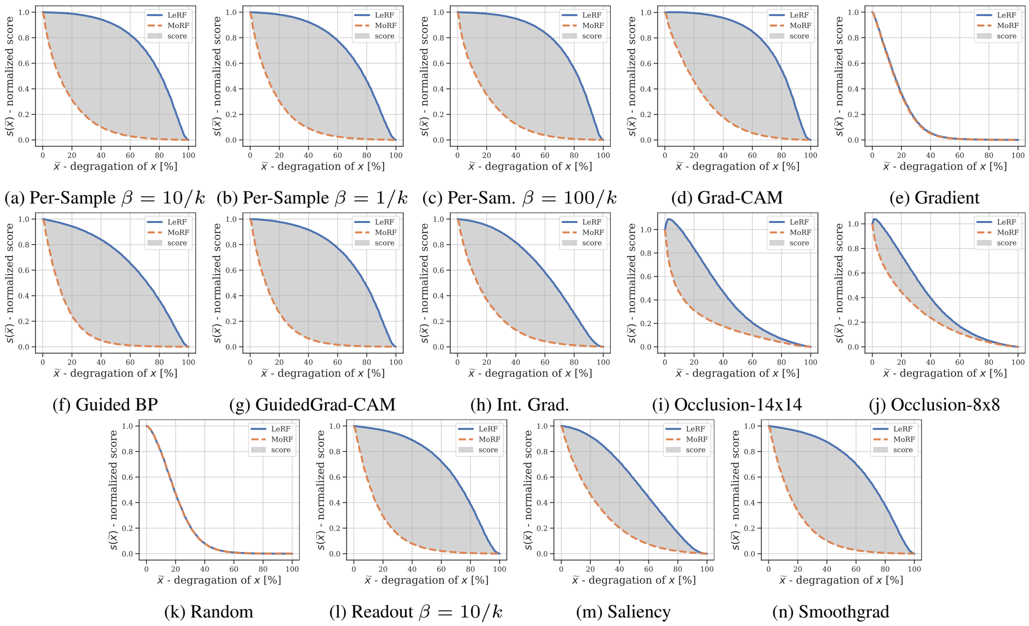


(a) Per-Sample $\beta = 10/k$ (b) Per-Sample $\beta = 1/k$ (c) Per-Sam. $\beta = 100/k$ (d) Grad-CAM (e) Gradient

(f) Guided BP (g) GuidedGrad-CAM (h) Int. Grad. (i) Occlusion-14x14 (j) Occlusion-8x8

(k) Random (l) Readout $\beta = 10/k$ (m) Saliency (n) Smoothgrad

Figure 8: MoRF and LeRF paths averaged over the ImageNet test dataset for the ResNet-50 network using 8x8 tiles.

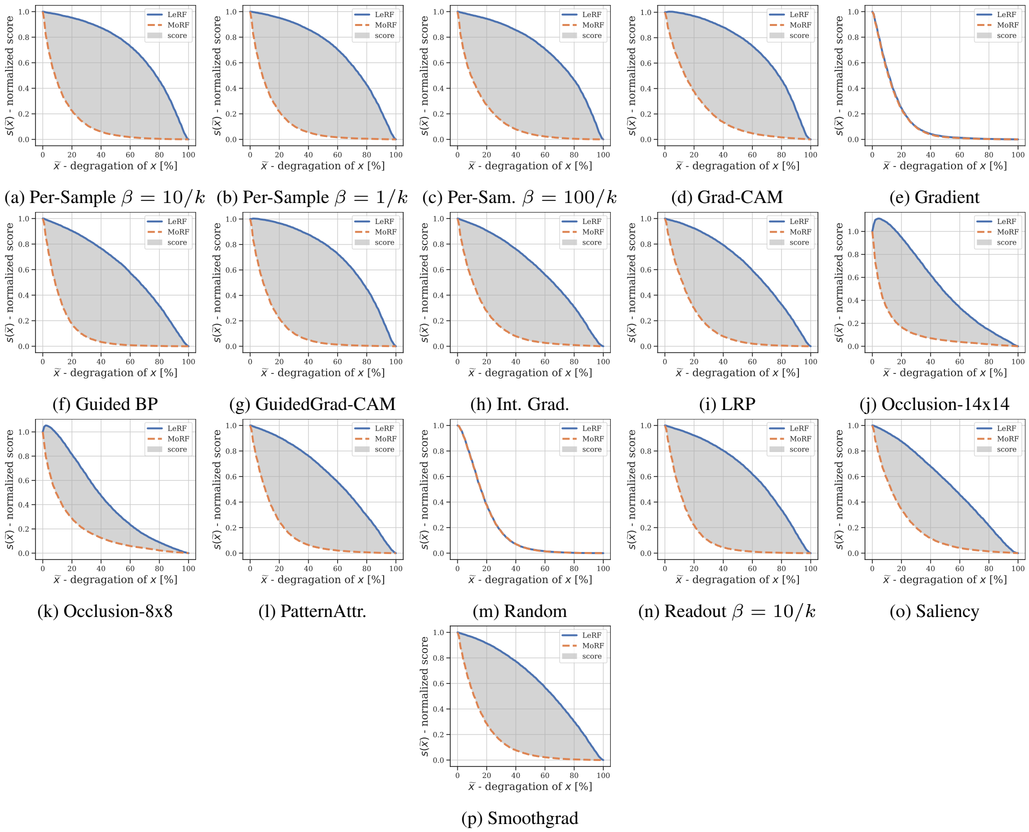


(a) Per-Sample $\beta = 10/k$ (b) Per-Sample $\beta = 1/k$ (c) Per-Sam. $\beta = 100/k$ (d) Grad-CAM (e) Gradient

(f) Guided BP (g) GuidedGrad-CAM (h) Int. Grad. (i) Occlusion-14x14 (j) Occlusion-8x8

(k) Random (l) Readout $\beta = 10/k$ (m) Saliency (n) Smoothgrad

Figure 9: MoRF and LeRF paths averaged over the ImageNet test dataset for the ResNet-50 network using 14x14 tiles.



(a) Per-Sample $\beta = 10/k$ (b) Per-Sample $\beta = 1/k$ (c) Per-Sam. $\beta = 100/k$ (d) Grad-CAM (e) Gradient

(f) Guided BP (g) GuidedGrad-CAM (h) Int. Grad. (i) LRP (j) Occlusion-14x14

(k) Occlusion-8x8 (l) PatternAttr. (m) Random (n) Readout $\beta = 10/k$ (o) Saliency

(p) Smoothgrad

Figure 10: MoRF and LeRF paths averaged over the ImageNet test dataset for the VGG-16 network using 14x14 tiles.